ADecimo: Model Selection for Time Series Anomaly Detection

Paul Boniol, Inria, ENS, CNRS, boniol.paul@inria.fr
Emmanouil Sylligardos, Inria, ENS, CNRS, emmanouil.sylligardos@ens.fr
John Paparrizos, The Ohio State University, paparrizos.1@osu.edu
Panos Trahanias, ICS-FORTH, trahania@ics.forth.gr
Themis Palpanas, Université Paris Cité & IUF, themis@mi.parisdescartes.fr

***Abstract*—Anomaly detection is a fundamental task for time-series analytics with important implications for the downstream performance of many applications. Despite increasing academic interest and the large number of methods proposed in the literature, recent benchmark and evaluation studies demonstrated that there exists no single best anomaly detection method when applied to heterogeneous time series datasets. Therefore, the only scalable and viable solution to solve anomaly detection over very different time series collected from diverse domains is to propose a model selection method that will choose, based on time series characteristics, the best anomaly detection method to run. This paper describes ADecimo, a modular and extensible web application that helps users understand the performance of time series classification algorithms used as model selection methods for time series anomaly detection. Overall, our system enables users to compare 17 different classifiers over 1980 time series, and decide on the most suitable time series classification method for their own time series and use cases.**

I. Introduction

Extensive collections of time-dependent measurements have become a reality in virtually every domain [1]. These measurements result in an ordered sequence of real-valued data points commonly called *time series*. Managing and analyzing time series collections is becoming important in multiple scientific and industrial applications from various domains, such as astronomy [2], energy sciences [3], and social sciences [4]. Anomaly detection, a particular type of time series analysis, has received ample academic and industrial attention [5], [6], [7], and has become an important problem that finds applications across a wide range of use cases. These applications share the same goal [8], [9], [10]: analyzing time series to identify observations that do not correspond to expected behavior. In practice, anomalies can correspond to [11]: (i) noise or erroneous data (e.g., faulty sensors); or (ii) actual data of interest (e.g., abnormal behavior of the system of interest). In both situations, detecting these anomalies is crucial for many applications.

Several recent surveys and experimental benchmarks describe and study the state-of-the-art time series anomaly detection methods [12], [5], [6], [13], [14]. Interestingly, these benchmark and evaluation studies demonstrated that no single anomaly detection method is the overall best when applied to time series originating from different domains. In practice, we observe that some approaches outperform others on time series with either specific characteristics (e.g., stationary or non-stationary time series), or containing specific types of anomalies (e.g., point-based or sequence-based anomalies). To overcome the above limitation, ensembling solutions have been proposed [15] that consist of running all existing anomaly detection methods and merging (e.g., averaging) all anomaly scores. Nevertheless, such solutions require executing all methods, thus, leading to running times that are prohibitive for large time series collections.

Therefore, the only scalable and viable solution to solve anomaly detection over time series from heterogeneous sources is to propose a model selection method that will select, based on time series characteristics, the best anomaly detection method to run. The model selection problem for anomaly detection in time series can be formalized as a time series classification problem (i.e., classifying time series into classes corresponding to the correct detectors to select). However, the lack of a benchmark with labeled time series has been a limiting factor for training robust model selection models; this only changed very recently [5], [6], [7].

Based on a very recent experimental evaluation of time series classification methods used as model selection approach for time series anomaly detection [16], we propose ADecimo[1], a system that aims to (i) easily visualize and assess the performances of times series classification methods used as model selection for times series anomaly detection, and (ii) allow the user to use pre-trained model selection method on their own data. ADecimo is based on the TSB-UAD benchmark [5] and employs time series from different domains and applications. Overall, the objective of ADecimo is to (i) facilitate the visualization and the understanding of the performances of model selection applied for time series anomaly detection, (ii) guide users in selecting appropriate approaches for their own use cases, and (iii) promote the development of AutoML methods [17] designed for model selection in the specific context of time series anomaly detection.

II. Preliminaries

We now provide the background necessary for the rest of the paper. We first briefly describe the pipeline used to compare time series classification methods as model selection approaches for time series anomaly detection. We then describe the datasets, anomaly detection methods, time series

[1]Available online: https://adecimots.streamlit.app/

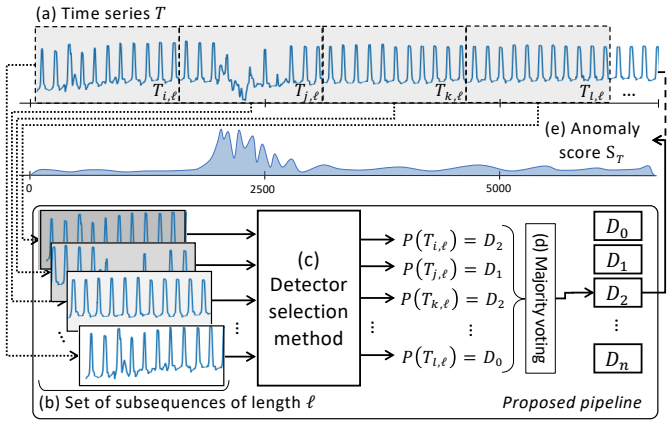


Fig. 1. Proposed pipeline for the method selection

classification approaches, and evaluation measures contained in our system.

A. Evaluation Pipeline

Time series classification can be performed with three strategies: (i) treating the entire time series as one sample, (ii) dividing the time series into overlapping subsequences, and (iii) dividing the time series into shifting subsequences (i.e., non-overlapping subsequences). We consider the third strategy for scalability reasons and to maximize the number of time series classification methods to be considered (not all methods can handle variable length time series).

Therefore, for all time series classification methods, we consider the experimental pipeline, illustrated in Figure 1, with the following steps: (i) **Preprocessing step**: Extraction of the subsequences of the same length ℓ (Figure 1(b)), (ii) **Prediction step**: Prediction of which detector to use for each subsequence (Figure 1(c)), and (iii) **Selection step**: Majority voting among all the different prediction to select one detector only (Figure 1(d)). The interested reader can find more technical details in [16].

B. Datasets and Methods

Datasets: We use the public datasets included in the TSB-UAD benchmark [5], comprising 18 datasets used in the literature, for a total of 1980 time series with labeled anomalies. Specifically, each point in every time series is labeled as normal or abnormal. In this demonstration, we consider 16 of the 18 datasets of TSB-UAD (described in Table I).

Anomaly Detection Methods: We select 12 different anomaly detection methods, summarized in Table I. Out of these, 8 are fully unsupervised (i.e., they require no prior information on the anomalies to be detected): IForest, IForest1, LOF, MP, NormA, PCA, HBOS, and POLY. The remaining 4 methods are semi-supervised (i.e., they require some information related to normal behaviors), namely, OCSVM, AE, LSTM-AD, and CNN. (The parameters for all methods are set as described in the TSB-UAD benchmark [5].)

Method Selection baselines: We then consider the method selection baselines summarized in Table I. We employ *feature-based* methods, that extract features using the tsfresh [18] library. We also use Rocket, a state-of-the-art *time series classifier*. Finally, we include two types of deep learning classifiers; (i) *Convolutional-based neural networks* and (ii) *Transformer-based neural networks* (all summarized in Table I). In total, we consider 16 methods, trained with windows lengths ℓ equal to 16, 32, 64, 128, 256, 512, 768, and 1024, for a total of 128 trained models.

Evaluation measures: We employ 4 evaluation measures. We use classification accuracy for model selection accuracy (i.e., the number of detectors correctly selected divided by the total number of time series). For anomaly detection accuracy, we use AUC-PR [19] and VUS-PR [13] (with a buffer length equal to 10 points). For execution time, we measure the *Training* time (i.e., the time required to train a model selection method), the *Selection* time (i.e., the time that a model needs to predict which model to use), and the *Detection* time (i.e., the time required to predict which detector to use and to run it).

TABLE I: Summary of datasets, methods, and measures.

Datasets	**Description**
Dodgers	unusual traffic after a Dodgers game **(1 time series)**
ECG	standard electrocardiogram dataset **(52 time series)**
IOPS	performance indicators of a machine **(58 time series)**
KDD21	composite dataset released in a recent SIGKDD 2021 **(250 time series)**
MGAB	Mackey-Glass time series with non-trivial anomalies **(10 time series)**
NAB	Web-related real-world and artificial time series **(58 time series)**
SensorScope	environmental data **(23 time series)**
YAHOO	time series based on Yahoo production systems **(367 time series)**
Daphnet	acceleration sensors on Parkinson's disease patients **(45 time series)**
GHL	Gasoil Heating Loop telemetry **(126 time series)**
Genesis	portable pick-and-place demonstrator **(6 time series)**
MITDB	ambulatory ECG recordings **(32 time series)**
OPPORTUNITY	motion sensors for human activity recognition **(465 time series)**
Occupancy	temperature, humidity, light, and CO2 of a room **(10 time series)**
SMD	Server Machine telemetry **(281 time series)**
SVDB	ECG recordings **(115 time series)**
Anomaly Detection	**Description**
IForest	constructs binary trees based on random space splitting. The nodes (i.e., subsequences) with shorter paths to the root are more likely to be anomalies.
IForest1	same as IForest, but each point (individually) are used as input.
LOF	computes the ratio of the neighboring density to the local density.
MP	detects abnormal subsequences with the largest nearest neighbor distance.
NormA	identifies normal patterns using clustering and calculates each subsequence weighted distance (with statistical criteria) to the normal patterns.
PCA	projects data to a lower-dimensional hyperplane, and data points with a significant distance from this plane can be identified as outliers.
AE	projects data to the lower-dimensional latent space and reconstructs the data, and outliers are expected to have larger reconstruction errors.
LSTM-AD	use an LSTM network that from the current subsequence tries to predict the following value. The error prediction is then used to identify anomalies.
POLY	fits a polynomial model that tries to predict the time series values from the previous subsequences. The outliers are detected with the prediction error.
CNN	builds, using a convolutional neural network, a correlation between current and previous subsequences. The anomaly score is the prediction deviation.
OCSVM	is a support vector method that fits the normal training dataset and finds the normal data's boundary.
HBOS	builds a histogram for the time series. The anomaly score is the inverse of the height of the bin.
Model Selection	**Description**
SVC	maps instances to points in space to maximize the gap between classes.
Bayes	uses Bayes' theorem to classify a point using class posterior probabilities.
MLP	consists of multiple layers of interconnected neurons
QDA	is a discriminant analysis algorithm for classification problems
AdaBoost	is a meta-algorithm using boosting technique with weak classifiers
Decision Tree	is an approach that splits data points into separate leaves based on features
Random Forest	is a set of Decision Trees fed with random samples and features.
kNN	assigns the most common class among its k nearest neighbors.
Rocket	transforms time series using a set of convolutional kernels, creating features used to train a linear classifier
ConvNet	uses convolutional layers to learn spatial features from the input data.
ResNet	is a ConvNet with residual connections between convolutional block
Inception Time	is a combination of ResNets with kernels of multiple sizes
SiT-conv	is a transformer architecture with a convolutional layer as input
SiT-linear	is a transformer architecture for which time series are divided into non-overlapping patches and linearly projected into the embedding space
SiT-stem	is a transformer architecture with convolutional layers with increasing dimensionality as input
SiT-stem-ReLU	is similar to SiT-stem but with Scaled ReLU.

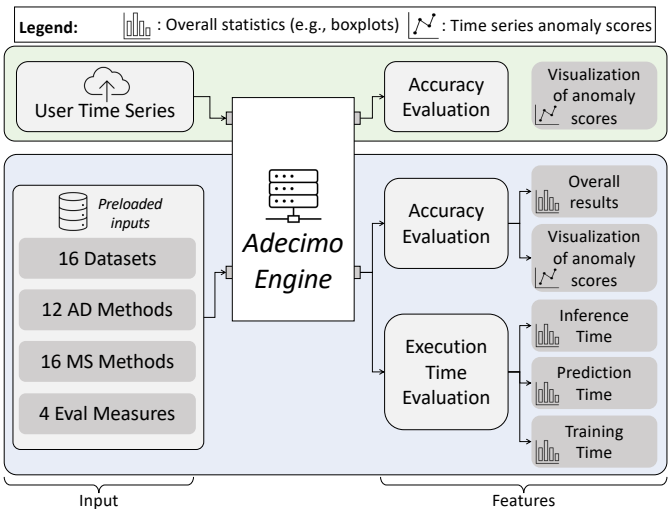


Fig. 2. Summary of our system inputs and features

III. ADecimo: System Overview

In this section, we describe ADecimo[2], the system we developed to help analysts understand the datasets, methods, and results of model selection approaches for time series anomaly detection. The GUI is a stand-alone web application developed using Python 3.6 and the Streamlit framework [20].

Figure 2 illustrates the inputs and features of ADecimo. The system is based on a preloaded set of datasets (16 in our demo), anomaly detection methods (12 in our demo), and accuracy evaluation measures (4 in our demo). The GUI permits interactions with these inputs. First, the user can visualize and interact with the overall experimental evaluation results (by filtering datasets or selecting only a subset of the methods). The user can also visualize the time series, the positions of the anomalies, and the model selection results. Finally, the user can upload their own time series and run our pre-trained model selection approaches.

The GUI is composed of three main frames, shown in Figure 3. The **Overall Accuracy** and the **Overall Execution time** frames contain aggregate results (Figure 3(A) for accuracy), the **Interactive Exploration** can be used to navigate the detailed evaluation results (Figure 3(B)), and the **Test on your own data** frame allow the user to use the pre-trained model to select a detector for their own time series (Figure 3(C)). In addition, a Description frame contains an overview of the system's objectives, and a Datasets and Methods frame lists information on the datasets and the methods in our evaluation.

We now describe in more detail the three main frames of the GUI and the corresponding available actions.

[Overall Accuracy Frame] The first frame depicts the overall accuracy evaluation and summarizes our results in a table (one accuracy value per time series and methods) and boxplots (as shown in Figure 3(A)). Using the sidebar on the left, the user can select which accuracy measure to use (i.e., VUS-PR or AUC-PR). Then, the user can filter based on datasets, families of methods (i.e., feature-based, convolutional-based, transformer-based, and rocket), and window length. The table and the boxplots (one per method) are updated based on the user's choices. Moreover, ADecimo provides a similar frame that summarizes: (i) training time, (ii) selection time, and (iii) detection time. The user also can choose to visualize the results on a linear or logarithmic scale.

[Interactive Exploration Frame] Finally, the user can click on a tab that opens the interactive Exploration frame (as shown in Figure 3(C)). In this frame, the user can visualize the chosen detector and the corresponding anomaly score for each time series and model selection method. For memory efficiency, ADecimo visualizes only snippets of up to 40k points. The anomaly scores of the detectors that have not been selected are also shown, but with a strong transparency ratio.

[Test on your own data Frame] Under the same interactive Exploration frame, the user can select (in the time series selection drop-down) the "Upload your own time series" option, which will run the selected model selection methods and print the chosen detector. ADecimo uses a bar-plot to visualize the distribution of the votes among all detectors. Finally, the uploaded time series is displayed, and the user can navigate through the time series interactively.

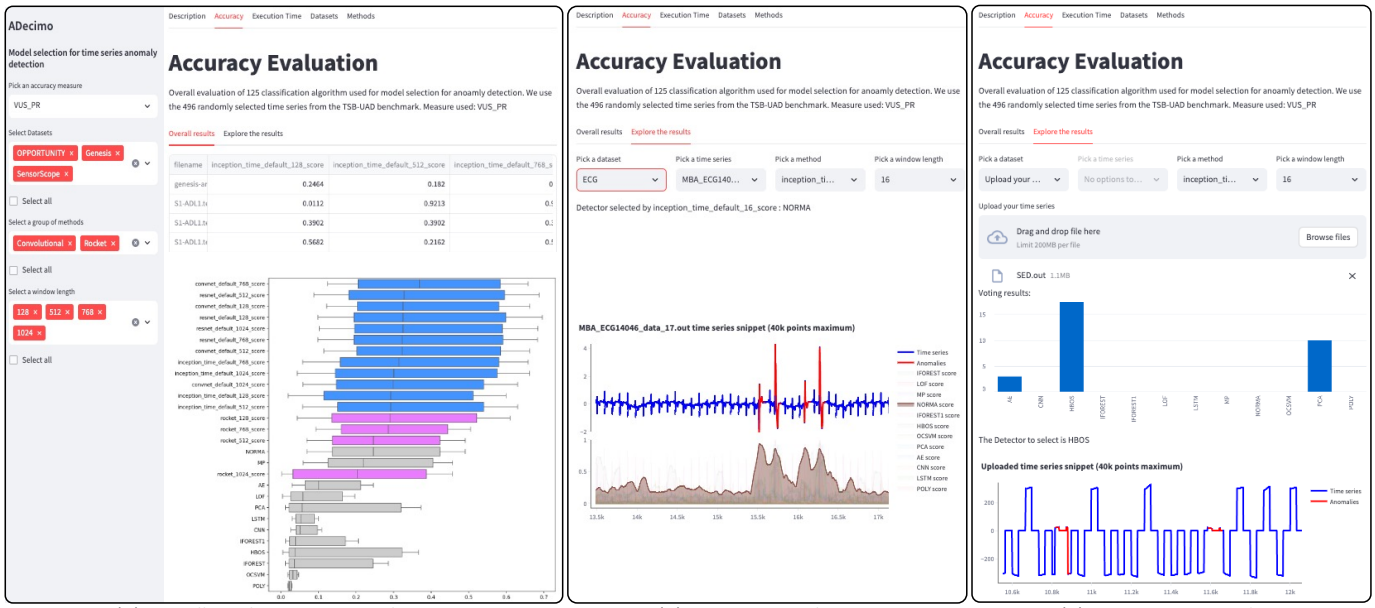
IV. Demonstration Scenarios

This demo has three goals: (i) showcase the importance of a webapp to dive into large experimental results and extract meaningful insight on how much performance a user can gain using model selection on a specific use case; (ii) enable the user to interactively visualize the model selection choices for specific individual time series, models and window length parameter; and (iii) challenge the user to test pre-trained model selection models on a new (or their own) time series.

[Scenario 1: Finding the best model selection method] This scenario starts in frames 1 and 2 (Figure 3(A)). Then, using the sidebar, we will ask the user to select datasets related to their application (e.g., medicine, environmental, or engineering). Then, the user can visit the *Datasets* frame to get more information on each dataset. Finally, the GUI will depict the most accurate (Figure 3(A)) and most scalable model selection methods (compared to existing anomaly detection methods) interactively. Thus, the user can discover if model selection methods outperform existing anomaly detection methods for a specific application, and if yes, which type of model selection approaches should be used.

[Scenario 2: Understanding and assessing model selection choice] In this scenario, we will ask the user to open the *Interactive Exploration* Frame (Figure 3(B)). In this frame, the user can select a specific dataset based on their application of interest and a model selection. The user will then be able to select each time series in the chosen datasets and visualize both the time series and the anomaly scores of the selected anomaly detection methods based on the chosen model selection approach. For instance, in Figure 3(B), the user selected one time series from the ECG dataset, and the InceptionTime method with a window length of 16. In this specific example, the user can see that the detector proposed

[2]Available online: https://adecimots.streamlit.app/



(A) Overall results: accuracy evaluation (B) Interactive evaluation (C) Test on your own data

Fig. 3. The three main ADecimo frames

by InceptionTime is NormA. As the GUI is also plotting the anomaly scores of all the remaining detectors, the user can assess if the choice made by the chosen model selection approach is correct. We will also explain the behavior of the model selection methods based on the methods' type and the window length impact.

[Scenario 3: Testing on your own data] In the last scenario, we will ask the user to click on the "upload your own" option in the time series selection drop-down. The user can add a new time series (like in Figure 3(C)). Our system will run the selected model selection approach and the chosen detector. The user will then evaluate the pertinence and accuracy of model selection approaches on the new data.

V. Conclusions

We demonstrate ADecimo, a system that enables users to understand our experimental evaluation of model selection for time series anomaly detection. It helps users discover if model selection can bring accuracy improvement for specific applications. Moreover, our systems allow the user to understand, assess, and test model selection on each time series of our benchmark and their own data.

References

[1] A. Bagnall, R. L. Cole, T. Palpanas, and K. Zoumpatianos, "Data Series Management (Dagstuhl Seminar 19282)," *Dagstuhl Reports*, vol. 9, no. 7, pp. 24–39, 2019.

[2] P. Huijse, P. A. Estevez, P. Protopapas, J. C. Principe, and P. Zegers, "Computational intelligence challenges and applications on large-scale astronomical time series databases," *IEEE Computational Intelligence Magazine*, vol. 9, no. 3, pp. 27–39, 2014.

[3] M. Bach-Andersen, B. Rømer-Odgaard, and O. Winther, "Flexible non-linear predictive models for large-scale wind turbine diagnostics," *Wind Energy*, vol. 20, no. 5, pp. 753–764, 2017.

[4] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. springer, 2016.

[5] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin, "Tsb-uad: An end-to-end benchmark suite for univariate time-series anomaly detection," *Proc. VLDB Endow.*, vol. 15, no. 8, 2022.

[6] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proc. VLDB Endow.*, vol. 15, no. 9, p. 1779–1797, jul 2022.

[7] E. Keogh, T. Dutta Roy, U. Naik, and A. Agrawal, "Multi-dataset Time-Series Anomaly Detection Competition 2021," https://compete.hexagon-ml.com/practice/competition/39/, 2021.

[8] V. Barnet and T. Lewis, *Outliers in Statistical Data*. John Wiley and Sons, Inc., 1994.

[9] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, ser. VLDB '06, 2006, p. 187–198.

[10] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *IEEE ICDM*, 2016, pp. 1317–1322.

[11] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier analysis*. Springer, 2017, pp. 1–34.

[12] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[13] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endow.*, vol. 15, no. 11, 2022.

[14] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: A benchmark for explainable anomaly detection over time series," *Proc. VLDB Endow.*, vol. 14, no. 11, p. 2613–2626, oct 2021.

[15] C. C. Aggarwal and S. Sathe, "Theoretical foundations and algorithms for outlier ensembles," *SIGKDD Explor. Newsl.*, vol. 17, no. 1, p. 24–47, sep 2015.

[16] E. Sylligardos, P. Boniol, J. Paparrizos, P. Trahanias, and T. Palpanas, "Choose wisely: An extensive evaluation of model selection for anomaly detection in time series," *Proc. of the VLDB Endow.*, vol. 16, no. 11, pp. 3418–3432, 2023.

[17] Y. Zhao, R. Rossi, and L. Akoglu, "Automatic unsupervised outlier model selection," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 4489–4502.

[18] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[19] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, 2006, p. 233–240.

[20] Streamlit documentation. https://streamlit.io/.